

# Enhancing a Mass Spectrometry I/O Framework

Eric Puryear,<sup>1</sup> David Angulo,<sup>2</sup> Kevin Drew,<sup>3</sup> Gregor von Laszewski,<sup>4</sup> Alex Schilling,<sup>5</sup>

<sup>[1]</sup> DePaul University, epuryear@students.depaul.edu

<sup>[2]</sup> DePaul University, dangulo@cs.depaul.edu

<sup>[3]</sup> The University of Chicago, kdrew@uchicago.edu

<sup>[4]</sup> Argonne National Laboratory, gregor@mcs.anl.gov

<sup>[5]</sup> The University of Illinois at Chicago, aschilli@uic.edu

## Abstract

*Managing the I/O and organization of mass spectrometry data in the form of files and structures within a program is often time consuming due to the many fields that need to be set and retrieved. Each programmer handling this in his/her own way leads to incompatibilities and inefficiencies between implementations. The Mass Spectrometry I/O Project addresses these issues by creating a framework that handles mass spectrometry data I/O and data organization thereby allowing researchers to concentrate on data analysis rather than I/O. In addition, the Mass Spectrometry I/O Project leverages several cross platform and portability enhancing technologies, allowing it to be utilized on a wide variety of hardware and operating system combinations.*

## 1 Introduction

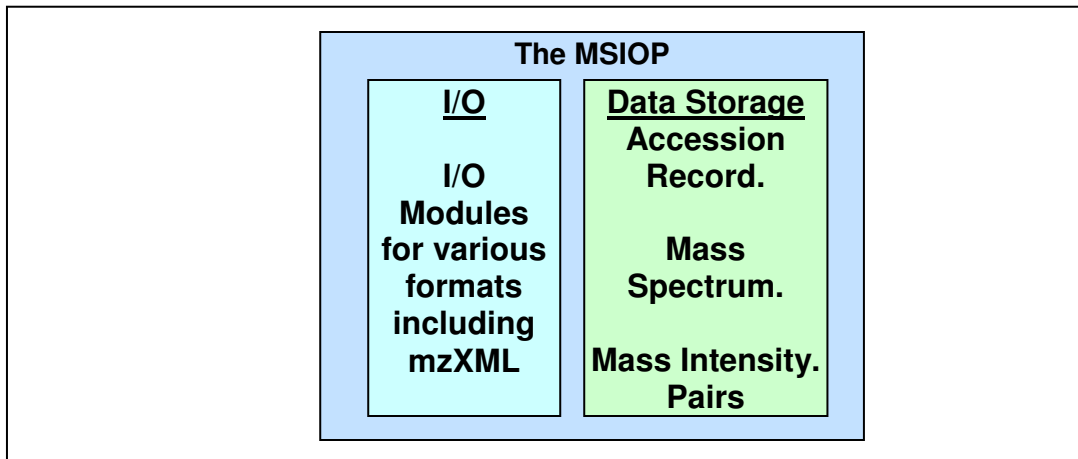
Mass spectrometers are used in proteomics to determine the mass/charge ratio and abundance of the various peptide fragments that comprise a particular peptide or protein [5]. The basic operation of a mass spectrometer involves placing a sample in the inlet, ionizing the sample, separating the ionized sample using methods such as electromagnetic fields, and detecting the separated particles. There are various types of mass spectrometers that use different ionization types such as matrix-assisted laser desorption/ionization (MALDI), and electrospray ionization (ESI), each of which takes a different approach to ionizing and separating the particles that comprise a sample. Each type of mass spectrometer is best suited for particular tasks as each method of ionization and detection has its own advantages and disadvantages. There is problem of incompatible data formats, and a lack of infrastructure to handle mass spectrometry data due to the use of proprietary file formats. With the ever increasing number of mass spectrometry data file formats, it becomes difficult and time consuming for programmers to handle these I/O and data management challenges [3]. The Mass Spectrum I/O Project (MSIOP) was designed to address these problems, allowing for the use of data from a multitude of mass spectrometers, while providing a sound infrastructure that includes input, output, and data type conversion, upon which mass spectrometry tools can be built [5].

Mass spectrometry is an increasingly important area of research within the field of bioinformatics. This problem was first addressed in a paper [5], and the modifications and improvements discussed in this paper further improve the functionality and performance of the Mass Spectrometry I/O Project (MSIOP) framework. This paper addresses the requirement and design considerations of the MSIOP, as well as the enhancements that were made and the functionality and performance results that were achieved.

The purpose of the MSIOP is to provide an I/O and data structure framework for mass spectrometry analysis tools and separate biologically significant programming from I/O. Handling the I/O and organization of data inside our framework and exposing a simple API allows researchers to spend their development time on code that accomplishes their research objectives.

To begin a discussion on the new version of the MSIOP, background information on the original MSIOP is needed. First envisioned in the spring of 2004 and implemented during the summer of that same year, the MSIOP was designed for use with mass spectrometry data from a variety of mass spectrometers and file formats. The MSIOP stores an abundance of metadata (data that describes the experiment's conditions) in addition to the peak lists, allowing researchers to annotate their data to facilitate the collaboration as well as curation of the data. The original MSIOP stores data in three structures. The top level structure is called Accession Record and contains information about the researcher, the mass spectrometer, and other metadata. The intermediate level structure is Mass Spectrum, which holds metadata that pertains to the lowest level structure, which is the Mass Intensity Pairs. The structure of the MSIOP is depicted in Figure 1.

The I/O portion of the MSIOP consists of modules for reading proprietary file formats, such as the .dta and .mgf formats. These modules are given a mass spectrometry data file, and create the appropriate MSIOP data storage structure.



**The Structure of the MSIOP,  
showing the three main  
structures and the I/O modules  
Figure 1**

Since it was first developed, the structure of the MSIOP has closely followed that of mzXML, a community standard for the storage of mass spectrometry data. The similarities between mzXML and the MSIOP allow for easy and efficient conversion between the MSIOP's internal data structures and an mzXML file. Figure 2 shows a small sample of an mzXML file which highlights similarities to the MSIOP. For example, the first ten lines are equivalent to part of an Accession Record in the MSIOP, while line eleven represents data that is stored in a Mass Spectrum. In an actual mzXML file, there would be an element named peak, which stores the peak list, and this element maps very closely to the Mass Intensity Pairs structure in the MSIOP.

<pre> 1. &lt;msManufacturer category="msManufacturer" value="Thermo" /&gt; 2. &lt;msModel category="msModel" value="LCQ Deca" /&gt; 3. &lt;msIonisation category="msIonisation" value="ESI" /&gt; 4. &lt;msMassAnalyzer category="msMassAnalyzer" value="Ion Trap" /&gt; 5. &lt;msDetector category="msDetector" value="EMT" /&gt; 6. &lt;software type="acquisition" name="Xcalibur" version="1.3" /&gt; 7. &lt;/msInstrument&gt; 8. &lt;dataProcessing centroided="1"&gt; 9. &lt;software type="conversion" name="IBG" version="1" /&gt; 10. &lt;/dataProcessing&gt; 11. &lt;scan num="1" msLevel="1" peaksCount="780" polarity="+" 12. retentionTime="PT130.12S" lowMz="410" highMz="1900" 13. basePeakMz="1727.36" basePeakIntensity="6.75227e+007" 14. totIonCurrent="5.90564e+008"&gt; </pre>	<div style="font-size: 3em; vertical-align: middle;">}</div>	Accession Record
<pre> 11. &lt;scan num="1" msLevel="1" peaksCount="780" polarity="+" 12. retentionTime="PT130.12S" lowMz="410" highMz="1900" 13. basePeakMz="1727.36" basePeakIntensity="6.75227e+007" 14. totIonCurrent="5.90564e+008"&gt; </pre>	<div style="font-size: 3em; vertical-align: middle;">}</div>	Mass Spectrum

**Data in the mzXML format  
Figure 2**

One of the most important features of the MSIOP is mzXML support. The mzXML format is quickly becoming a preferred method of storing mass spectrometry data. Because mzXML stores ample amounts of metadata that relates to the experiment's parameters as well as derived data, and is platform independent, mzXML is being integrated into an increasing number of mass spectrometry toolkits, including the MSIOP, especially since The Institute for Systems Biology began promoting the mzXML standard [4].

## 2 Requirements

Since the MSIOP is to function as a simple yet robust I/O and data structure framework, it must meet several requirements that address usability as well as functionality. The requirements for the MSIOP are as follows.

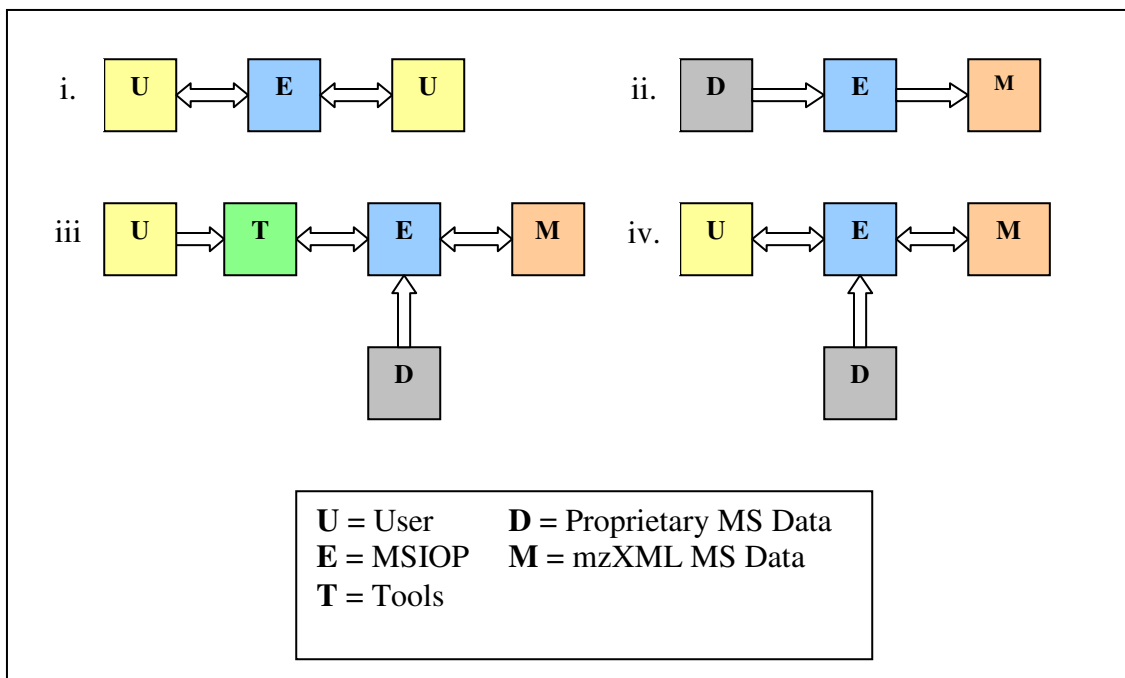
- The MSIOP should separate out I/O functionality from biologically significant programming logic.
- The MSIOP should provide programming routines/functions and abstractions common to many mass spectrometry analysis tools.

- The MSIOP should provide baseline infrastructure for rapid development of mass spectrometry analysis tools.
- The MSIOP should create flexible infrastructure that allows developers to abstract and hide details when they want, and also allow developers to use low level routines/functions when necessary.
- The MSIOP should allow and promote the reuse of code.
- The MSIOP should be portable to allow developers to utilize its framework on a variety of platforms.
- The MSIOP should efficiently handle large mass spectrometry files and data sets.

### 3 Use cases

For these use cases, the symbol **U** represents a user of the MSIOP and code that they have written, **E** represents the MSIOP, **T** represents an existing tool, **D** represents mass spectrometry data in any supported format, and **M** represents data in the mzXML format. These use cases are depicted in Figure 3.

- i. A user uses the MSIOP as a layer between one or more mass spectrometry tools, allowing for compatibility and data consistency.
- ii. A user leverages the MSIOP's ability to read and write from multiple formats to convert from proprietary formatted files to mzXML.
- iii. A user leverages an existing tool, such as SpectralMatch, which makes use of the MSIOP framework.
- iv. A user creates a mass spectrometry analysis tool that implements the MSIOP for I/O and data structures.

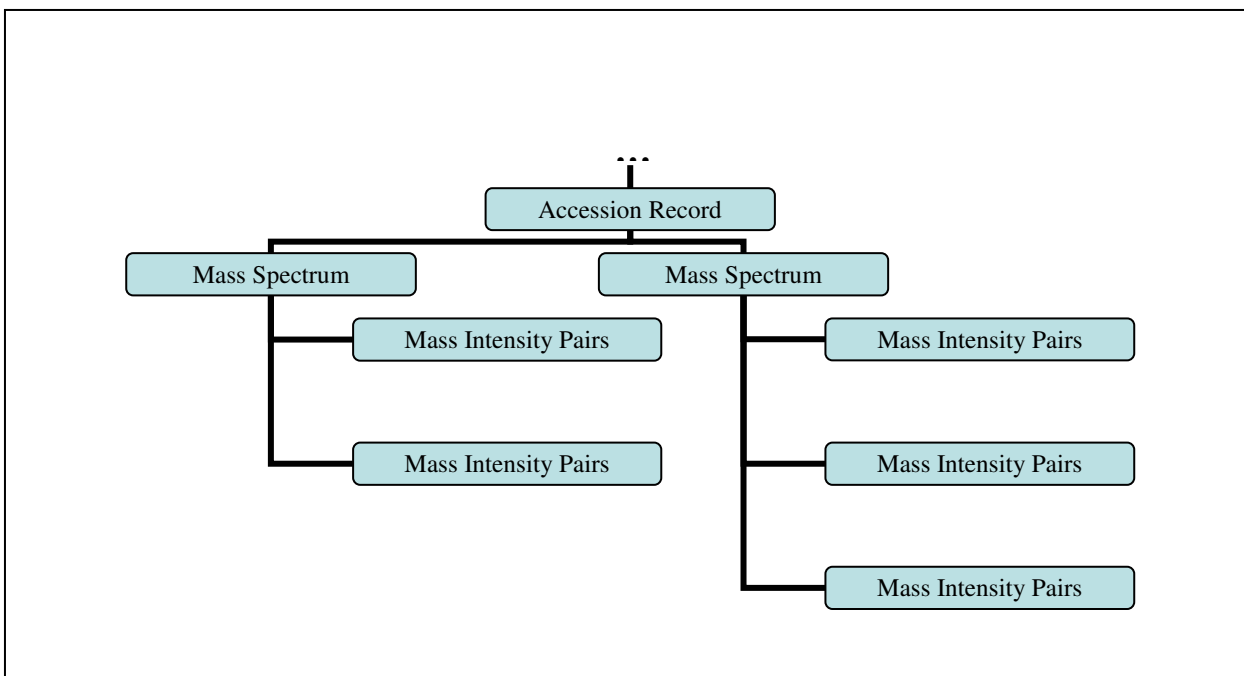


**Use cases for the MSIOP**  
**Figure 3**

A certain amount of overlap will be seen in the above use cases. For example, cases ii and iv will often occur simultaneously as users will convert between two or more file formats while also creating their own toolkit.

#### 4 Architecture

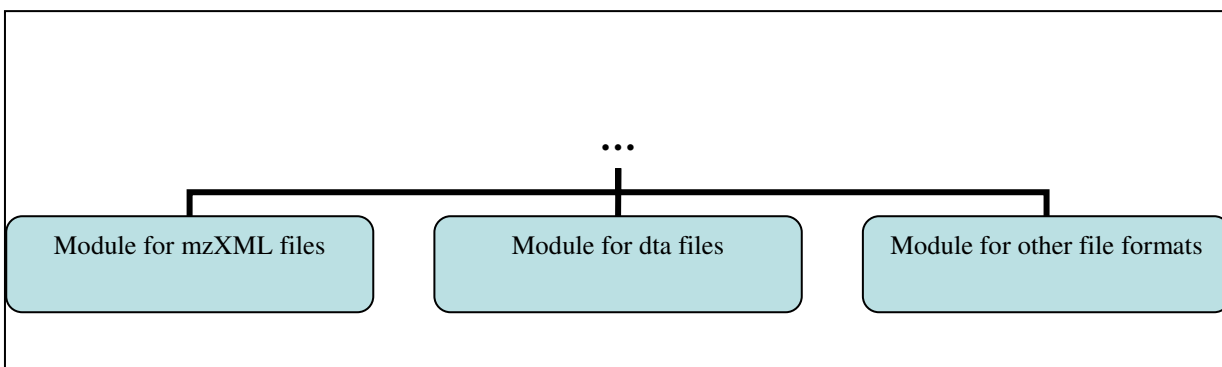
The MSIOP has three main structures as shown in Figure 1. These are the Accession Record, Mass Spectrum, and Mass Intensity Pairs, and they form a multilevel storage system for mass spectrometry data. Each Accession Record can have one or more child Mass Spectrum, and each Mass Spectrum can have one or more child Mass Spectrum, along with one or more child Mass Intensity Pairs. One possible example of this hierarchical layout can be seen in Figure 4.



### The Hierarchy of the MSIOP's Data Structures

The I/O modules, as shown in Figure 5, are responsible for reading data into the structures shown in Figure 4, and for writing data from the structures to the open mzXML format, as well as manufacturer specific mass spectrometry data file formats.

The data structures and I/O modules that comprise the MSIOP form a framework, allowing mass spectrometry researchers and software developers to use one unified API to handle the I/O as well as access to the data structures. Furthermore, the use of a single API allows changes to the underlying structure and implementation of the MSIOP without affecting the functionality of the MSIOP or any software constructed upon the MSIOP framework.



### The Hierarchy of the MSIOP's I/O Modules

## 5 Implementation

The MSIOP exposes a single cohesive API to developers, allowing them to easily read, write, and manipulate mass spectrometry data that is in a file or the MSIOP's structures. Selected samples of pseudo code for the MSIOP can be seen in Figures 6 through 9.

The pseudo code in Figures 6 through 9 demonstrates a small portion of the MSIOP. Figure 6 depicts the instantiation of each of the three main data structures of the MSIOP. Figure 7 demonstrates how these structures are linked together in a hierarchical fashion, while figure 8 shows the pseudo code for accessing and modifying the data held within these structures. Figure 9 shows the function calls that are responsible for file I/O.

The MSIOP is implemented in the C programming language. This language was chosen for speed and portability. Since portability is a major goal of the MSIOP, every effort has been made to produce code that will function on a variety of platforms. In keeping with this goal, the MSIOP is able to check what the endian type of the system is that it is running on and adapt to this type.

```
//Functions for creating the MSIOP's Data Structures

AccessionRecordPointer createAccessionRecord ()
    //Creates an Accession Record structure and returns a pointer.

MassSpectrumPointer createMassSpectrum ()
    //Creates a Mass Spectrum structure and returns a pointer.

MassIntensityPairsPointer createMassIntensityPairs ()
    //Creates a Mass Intensity Pairs structure and returns a pointer.
```

**Pseudo code for instantiating the MSIOP data structures**  
**Figure 6**

```
//Functions for linking the MSIOP's Data Structures Together

void appendMassSpectrumToAccessionRecord (AccessionRecordPointer,
                                           MassSpectrumPointer)
    //Appends a Mass Spectrum to an Accession Record

void appendMassSpectrumToMassSpectrum (MassSpectrumPointer,
                                        MassSpectrumPointer)
    //Appends a Mass Spectrum to an Accession Record

void appendMassIntensityPairsToMassSpectrum (MassSpectrumPointer,
                                             MassIntensityPairsPointer)
    //Appends Mass Intensity Pairs to a Mass Spectrum
```

**Pseudo code for connecting the MSIOP data structures**  
**Figure 7**

```

//Functions for accessing and mutating data in the MSIOP's Data Structures

void setMsMassAnalyzer (AccessionRecordPointer, msMassAnalyzerEnum)
    //Sets the type of mass analyzer used in the experiment

void setMass (MassIntensityPairsPointer, mass)
    //Sets the mass of individual Mass Intensity Pairs

double getMass(MassIntensityPairsPointer)
    //Retrieves the mass of individual Mass Intensity Pairs

```

**Pseudo code for accessing and mutating data in the MSIOP data structures**  
**Figure 8**

```

//Functions for file I/O

AccessionRecordPointer read_mzXML (mzXML file)
    //reads an mzXML file and creates an Accession Record that represents the file.

MassSpectrumPointer read_mgf (mgf file)
    //reads an mgf file and creates a Mass Spectrum that represents the file.

MassSpectrumPointer read_dta (dta file)
    //reads a dta file and creates a Mass Spectrum that represents the file.

void write_mzXML (AccessionRecordPointer)
    //reads an Accession Record and writes the data to an mzXML file.

```

**Pseudo code for file I/O**  
**Figure 9**

Due to the high computational and storage demands of mass spectrometry analysis tools, the MSIOP has been designed to deliver a high level of performance. This is accomplished by the use of libxml2, known to be a very fast and efficient parser [8], for mzXML I/O.

The libxml2 library is used because of its high compliance with xml standards, flawless performance on a variety of compliance tests designed by the WC3 [8], and performance based on the most recent comparisons. In a comparison against a series of other xml libraries, libxml2 consistently used less memory, validated and parsed faster, and wrote xml documents in less time than the alternative libraries [8]. The mzXML reader validates and parses an mzXML file and creates an Accession Record with the appropriate children Mass Spectrum and Mass Intensity Pairs. The mzXML writer takes an Accession Record, traverses the hierarchical structure, and creates an mzXML file that represents the Accession Record and its substructures.

The MSIOP is available in the form of shared libraries for Linux. This allows developers to have one instance of the MSIOP per computer, allowing for easy upgrading when new versions of the MSIOP are released. These shared libraries can also be wrapped with SWIG. SWIG (Simplified Wrapper and Interface Generator) [1] allows for

the creation of wrappers that easily integrate C code with other languages such as Java, Perl, and Python. With SWIG wrappers around the MSIOP's shared libraries, it is possible to quickly and easily make use of the entire MSIOP framework in a few lines of Perl, Python, or Java code.

Despite the modifications made to the MSIOP, which range from the improved support of various file formats and changes to the underlying infrastructure, the current version of the MSIOP is compatible with previous versions. In addition, tools that leverage the previous MSIOP framework will see an improvement in performance and reliability by upgrading to the current MSIOP. These performance gains are the result of optimizations to the underlying data structures and functions.

## 6 Results

The performance of the MSIOP has been evaluated and found to be satisfactory for the purpose that it was envisioned. Table 1 shows read and write speed performance statistics for the mzXML I/O modules when running on an Intel Pentium 4 3.2GHz processor with 2GB of RAM under Linux 2.4 with a 7200 RPM ATA100 Hard Drive. Since competing pluggable mzXML I/O modules do not exist, other timings are not available for comparison; however the read speed is consistent with what can be expected from the hardware utilized, and the write speed will be improved in future versions.

Read Or Write	User Time	System Time	File Size (KB)	KB/s
Read	0.87s	0.090s	14216K	16340
Write	7.46s	0.10s	14216K	1905

**Table 1**  
**Performance results for the MSIOP mzXML I/O Modules**

The MSIOP is currently utilized by several tools. These include Spectral Comparison, HDXRates, and SpectralMatch [insert citation]. Each of these tools uses the MSIOP framework to handle file I/O and internal data management.

## 7 Conclusion

The MSIOP is a robust framework upon which mass spectrometry analysis tool can be built. With its ability to handle mzXML files as well as proprietary file format, the MSIOP separates biologically significant programming from I/O, allowing developers to quickly construct mass spectrometry analysis tools, or integrate file I/O into existing tools. By using the SWIG shared libraries, the MSIOP can be used by a variety of other programming languages, including Java, Python, and Perl, further extending the MSIOP's utility.

## Acknowledgements

This research was done by members of the Illinois Biogrid, in conjunction with Argonne National Laboratory. This material is based upon work supported by the National Science Foundation under Grant No. 0353989.

## References

- [1] Beazley, David M. *Tcl and SWIG as a C/C++ Development Tool*. Department of Computer Science, University of Chicago. 1998.
- [2] Desiere, Frank. *Integration with the human genome of peptide sequences obtained by high-throughput mass spectrometry*. *Genome Biology*, volume 6, issue 1, pages 6-18, September 2004.
- [3] Drew, Kevin; Angulo, David; Schilling, Alex; Freeman, Tim. *Mass Spectra Analysis on the Illinois Biogrid*. Proceedings of 2004 Midwest Software Engineering Conference, Chicago. June 2004. (Poster). (Electronically published <http://facweb.cs.depaul.edu/bioinformatics/publications/MSEC-MassSpec-Poster2004.ppt>).
- [4] Pedrioli, Patrick G. *A common open representation of mass spectrometry data and its application to proteomics research*. *Nature Biotechnology*. Volume 22, issue 11, pages 81-92, November 2004.
- [5] Puryear, Eric; Angulo, David; Drew, Kevin; Schilling, Alex; von Laszewski, Gregor. *Developing a Distributed and Scalable Foundation for Mass Spectrometry Data*. DePaul University CTI Tech Report, pages 1-7. April 2005. (To be published)
- [6] Rusconi, F.; Belghazi, M. *Desktop Prediction/Analysis of Mass Spectrometric Data in Proteomic Projects by using MassXpert*. *Bioinformatics* 2002, Volume 18, issue 4, 2002.
- [7] Steele, Adam; Angulo, David. *The Illinois BioGrid: A Prototype for Industry-Academe Collaboration*. Proceedings of 2003 Midwest Software Engineering Conference, Chicago. June 2003(Poster). (Electronically published <http://facweb.cs.depaul.edu/bioinformatics/Publications.htm>).
- [8] XML Speed Comparison. Web Page. Available from: <http://xmlbench.sourceforge.net/results/benchmark200402/>